

FIG.1

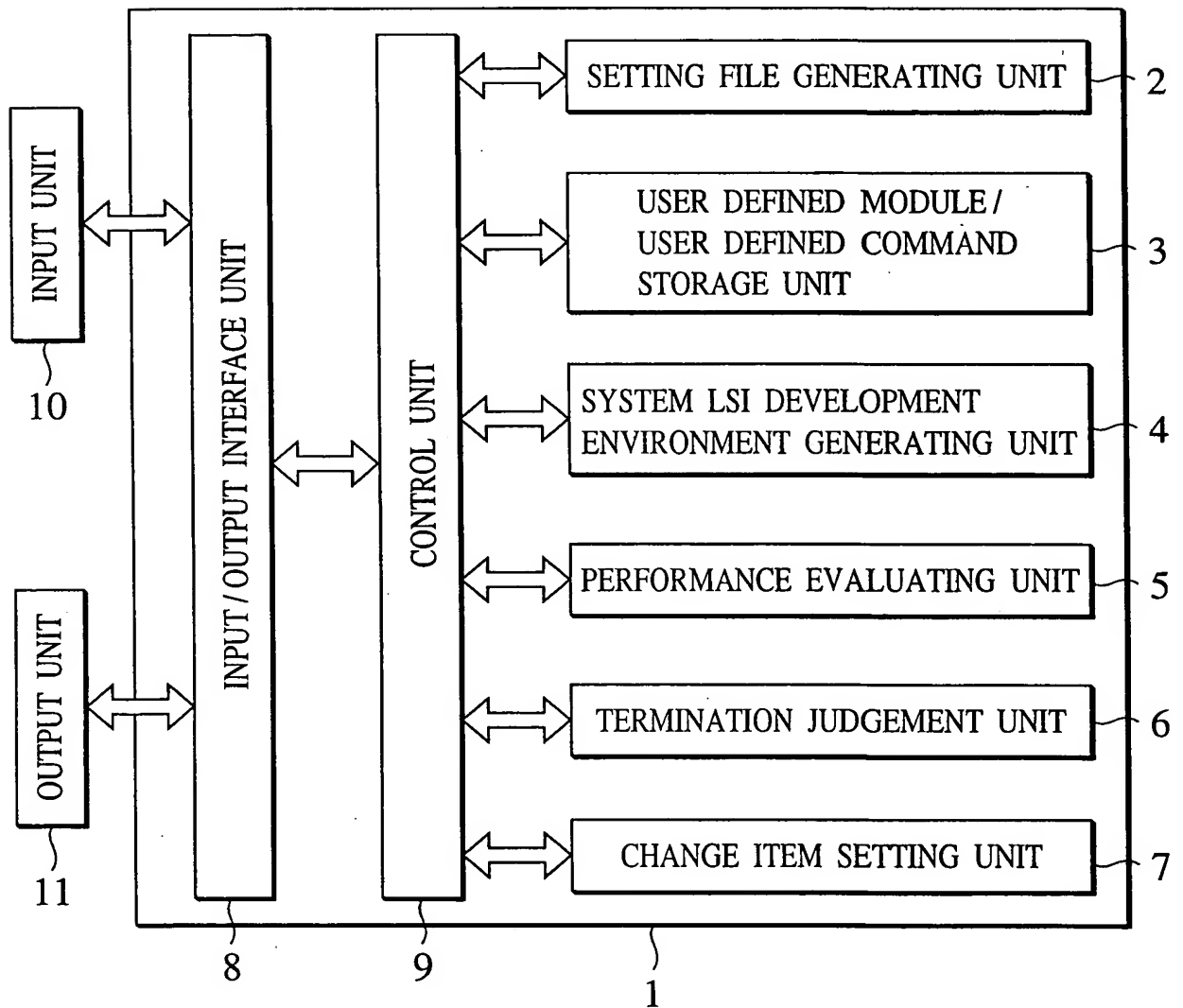


FIG.2

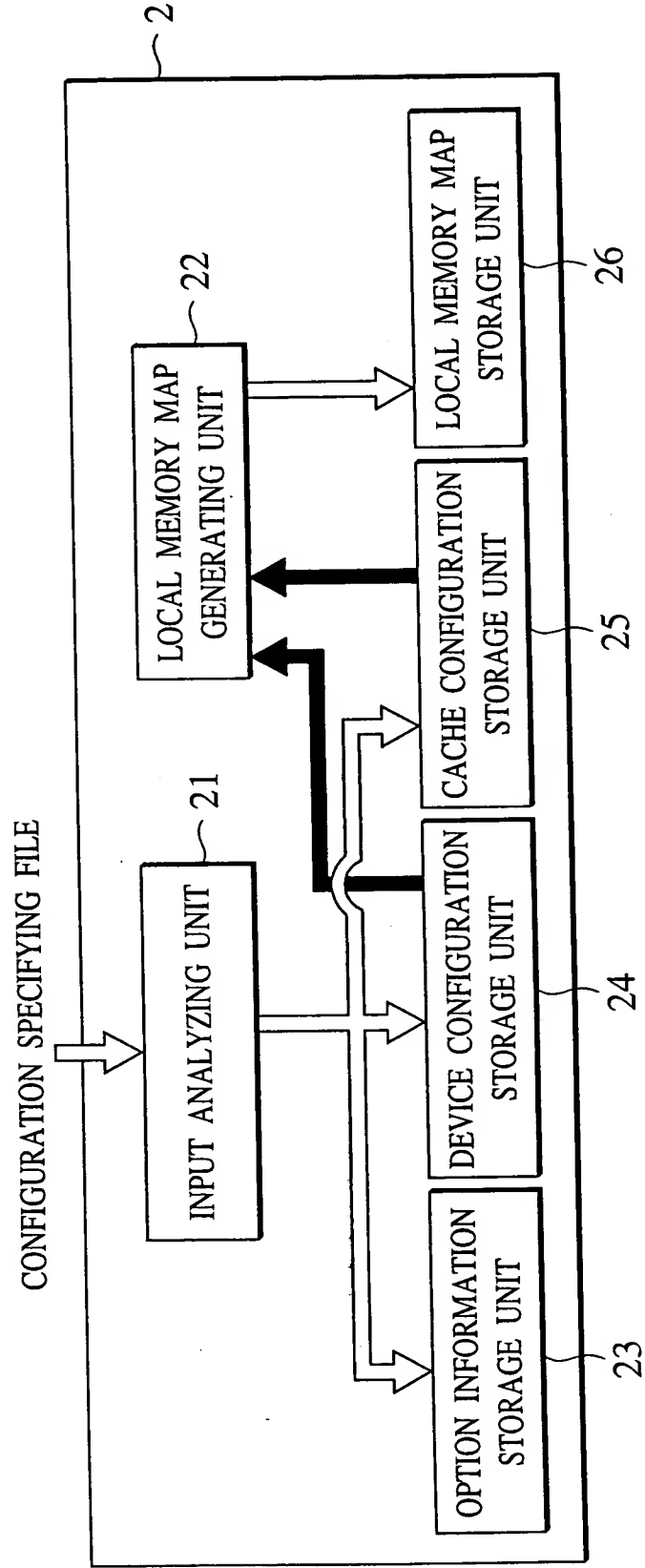


FIG.3

```
//configuration data
CHIP_NAME="SimpleChip1"//CHIP NAME
FREQUENCY=200; //FREQUENCY UNIT.MHz,
//PROCESSOR1
P_MODULE[Processor1](
    P_ENGINE(
        //SPECIFICATION OF PROCESSOR CORE
        CORE{
            ID=0; //CORE ID
            //OPTION COMMAND
            DIV=ON; //DIVIDING COMMAND
            MINMAX=ON; //MAXIMUM / MINIMUM VALUE COMMAND
            BIT=ON; //BIT OPERATION COMMAND
        };
        IMEM{
            SIZE=4; //KB
        };
        DMEM{
            SIZE=2; //KB
        };
        BIU{ //BUS INTERFACE COMMAND
            BUS_SIZE=64;
        };
        INTC{ //INTERRUPT CONTROLLER
            CHANNEL_BITW=16; //NOT DEFINED
            LEVEL=15; //NOT DEFINED
        };
        DSU{ //DEBUG UNIT
            INST_ADDR_BREAK_CHANNEL=1;
            DATA_ADDR_BREAK_CHANNEL=1;
        };
    );
);
//PROCESSOR2
P_MODULE[Processor2](
    P_ENGINE(
        CORE{
            ID=1; //CORE ID
            //OPTION COMMAND
            DIV=ON; //DIVIDING COMMAND
            MINMAX=OFF; //MAXIMUM / MINIMUM VALUE COMMAND
            BIT=OFF; //BIT OPERATION COMMAND
        };
        IMEM{ //COMMAND RAM
            SIZE=8; //KB
        };
        DMEM{ //DATA RAM
            SIZE=20; //KB
        };
        ICACHA{ //COMMAND CACHE
            SIZE=4; //KB
        };
        DCACHE{ //DATA CACHE
            SIZE=8; //KB
        };
    );
    //COPROCESSOR
    COPRO[Cop1]{
        IS_VLIW=YES; //VLIW TYPE COPROCESSOR
        VLIW_BTW=32; //COMMAND LENGTH 32 BITS
        ISA_DEFINE="cop1.isa"; //ISA DEFINITION, ANOTHER FILE
        RTL="cop1.v" //RTL DESCRIPTION, ANOTHER FILE
    };
);
//USER DEFINITION
UCI{
    ISA_DEFINE="uxi1.uci"; //ISA DEFINITION, ANOTHER FILE
    RTL="uxi1.v"; //RTL DESCRIPTION, ANOTHER FILE
    SIM="ucimodel.c"; //HARDWARE C MODEL, ANOTHER FILE
};
);
//GLOBAL MAP
GLOBAL_MAP=:schip1.map; //SPECIFY ANOTHER FILE
//end of file
```

FIG.4A

```
//start : size : name : cache(opt) : shadow_original_start(opt) : scope : type : read_write(opt)
0X0000_0000 : 0X0020_0000 : RAM1 :      : : global : memory : ro ;
0X0080_0000 : 0X0080_0000 : RAM2 : Cache : : global : memory : rw ;
0X8080_0000 : 0X0080_0000 : RAM3 :      : : global : memory : rw ;
```

FIG.4B

```
:RAM1      :0X00000000 : 0X00200000 :      : :global : ro::      memory:
:Imem0     :0X00200000 : 0X00002000 :      : : local : rw::      Imem:   in mm
:Imem1     :0X00202000 : 0X00002000 :      : : local : rw::      Imem:   in mm
:dmem0     :0X00208000 : 0X00004000 :      : : local : rw::      dmem:   in mm
:dmem1     :0X0020c000 : 0X00004000 :      : : local : rw::      dmem:   in mm
:icache_dat :0X00300000 : 0X00004000 :      : : local : rw:: icache_data: in mm
:icache_tag :0X00310000 : 0X00004000 :      : : local : rw:: Icache_tag: in mm
:RAM2      :0X00800000 : 0X00800000 : cache : :global : rw::      memory:
:RAM3      :0X00808000 : 0X00800000 :      : :global : rw::      memory:
```

FIG.4C

DECLARING COMMAND MNEMONIC, OPERATION CODE, AUGMENT

```
void xor(int_reg_modify,int_reg_arc);
{
    code16="0000_0010_0000" ;
}
short xori(int_reg_src,signed char_imm);
{
    code16="0000_0011_iiii_iiii";
}
```

FIG.5

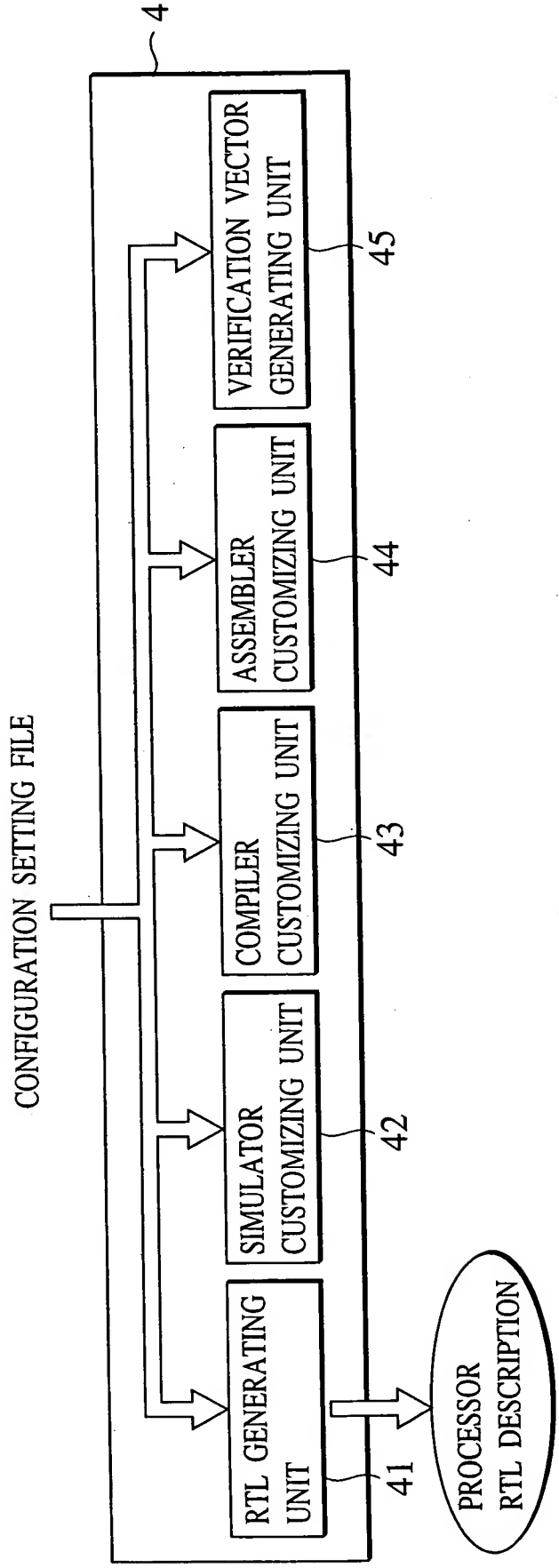


FIG.6

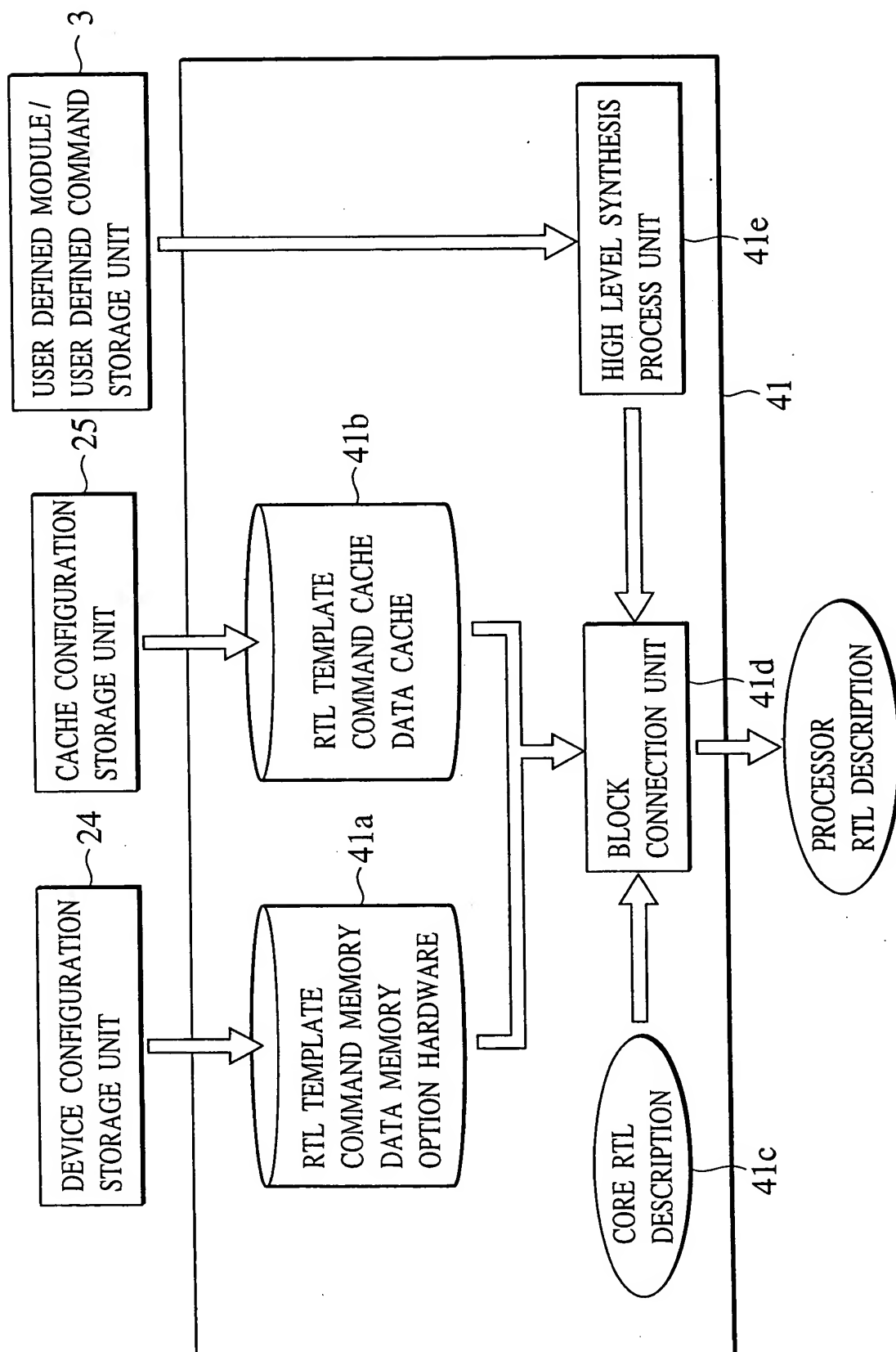


FIG. 7

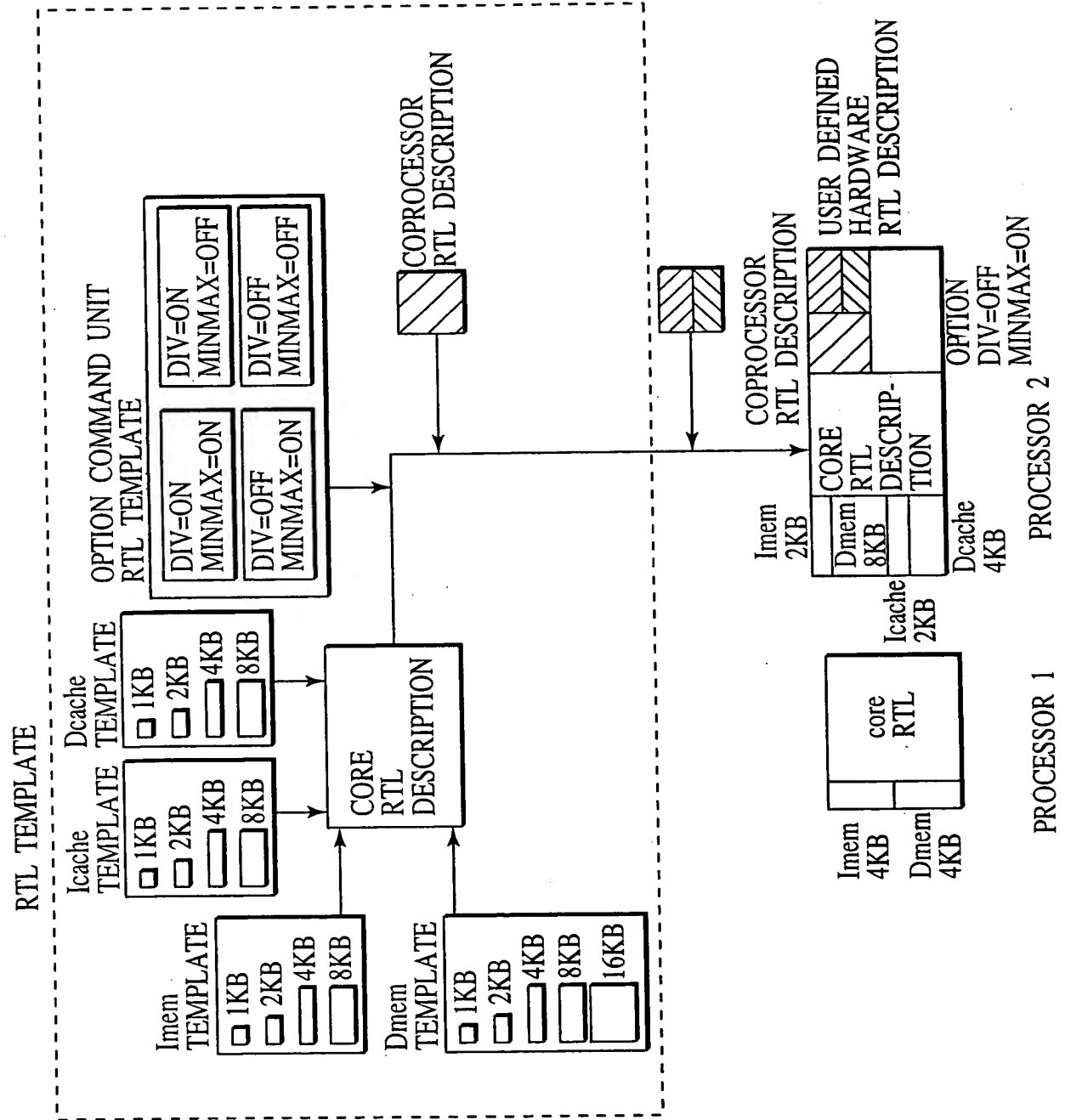


FIG.8

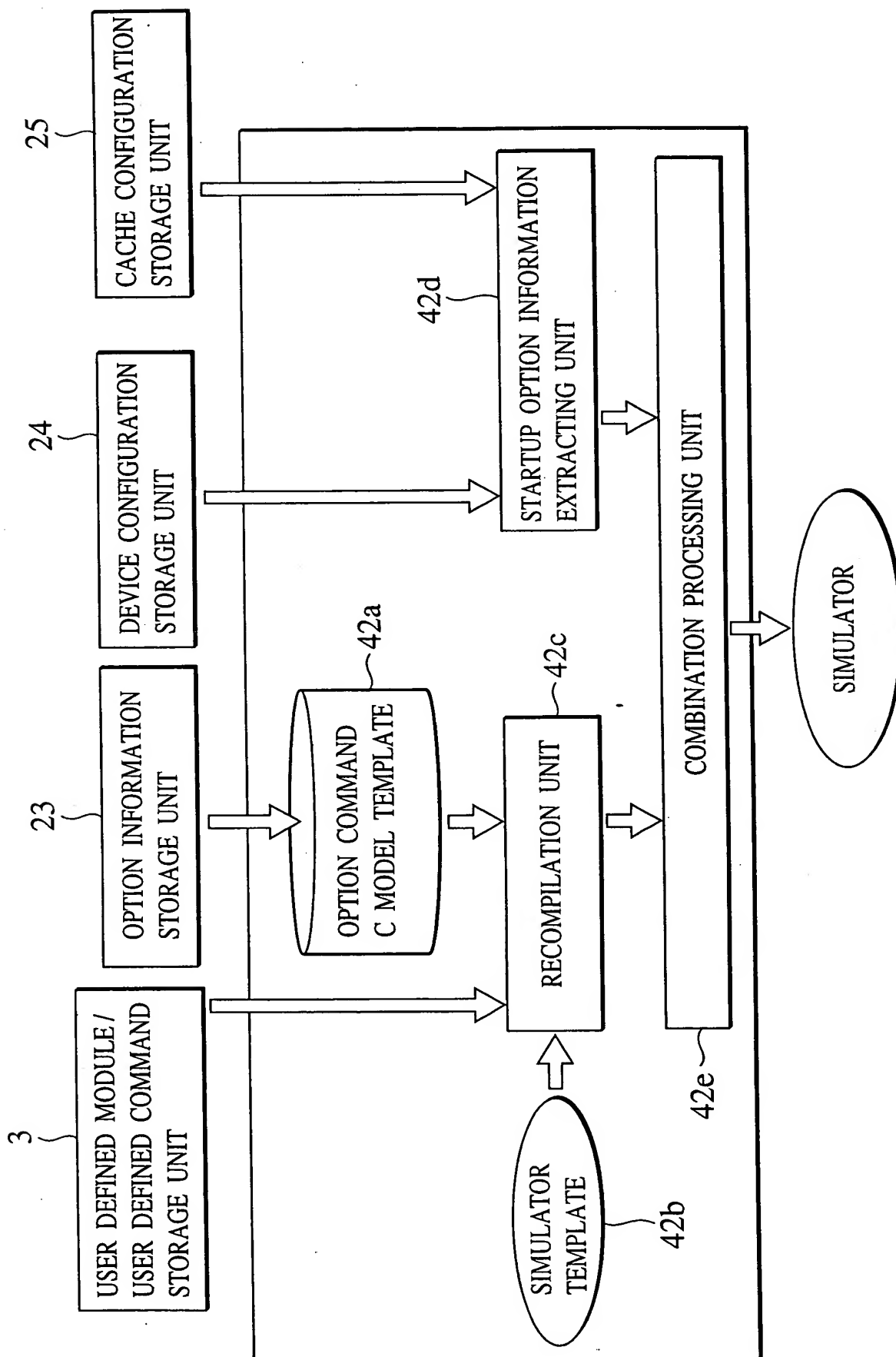


FIG.9A

```
-opt_minmax=ON  
-opt_div=ON  
-opt_bit=OFF  
-rom=0x00000000,0x00200000  
-imem=0x00200000,0x00300800  
-dmem=0x00300000,0x00400000  
-icache=0x00400000,0x007f0000  
-dcache=0x007f0000,0x00800000  
-ram=0x00800000,0x01000000  
-ram-shadow=0x80800000,0x81000000  
⋮
```

FIG.9B

```
-rom=0x00000000,0x00200000  
-imem=0x00200000,0x00300800  
-dmem=0x00300000,0x00400000  
-icache=0x00400000,0x007f0000  
-dcache=0x007f0000,0x00800000  
-ram=0x00800000,0x01000000  
-ram-shadow=0x80800000,0x81000000  
⋮
```

FIG.9C

```
// OPTION COMMAND  
void_asm min(int_reg_modify,int_reg_src);  
void_asm max(int_reg_modify,int_reg_src);  
// USER DEFINED COMMAND  
void_asm xor(int_reg_modify,int_reg_src);  
short_asm xori(int_reg_src,signed char_imm);  
⋮
```

FIG.10

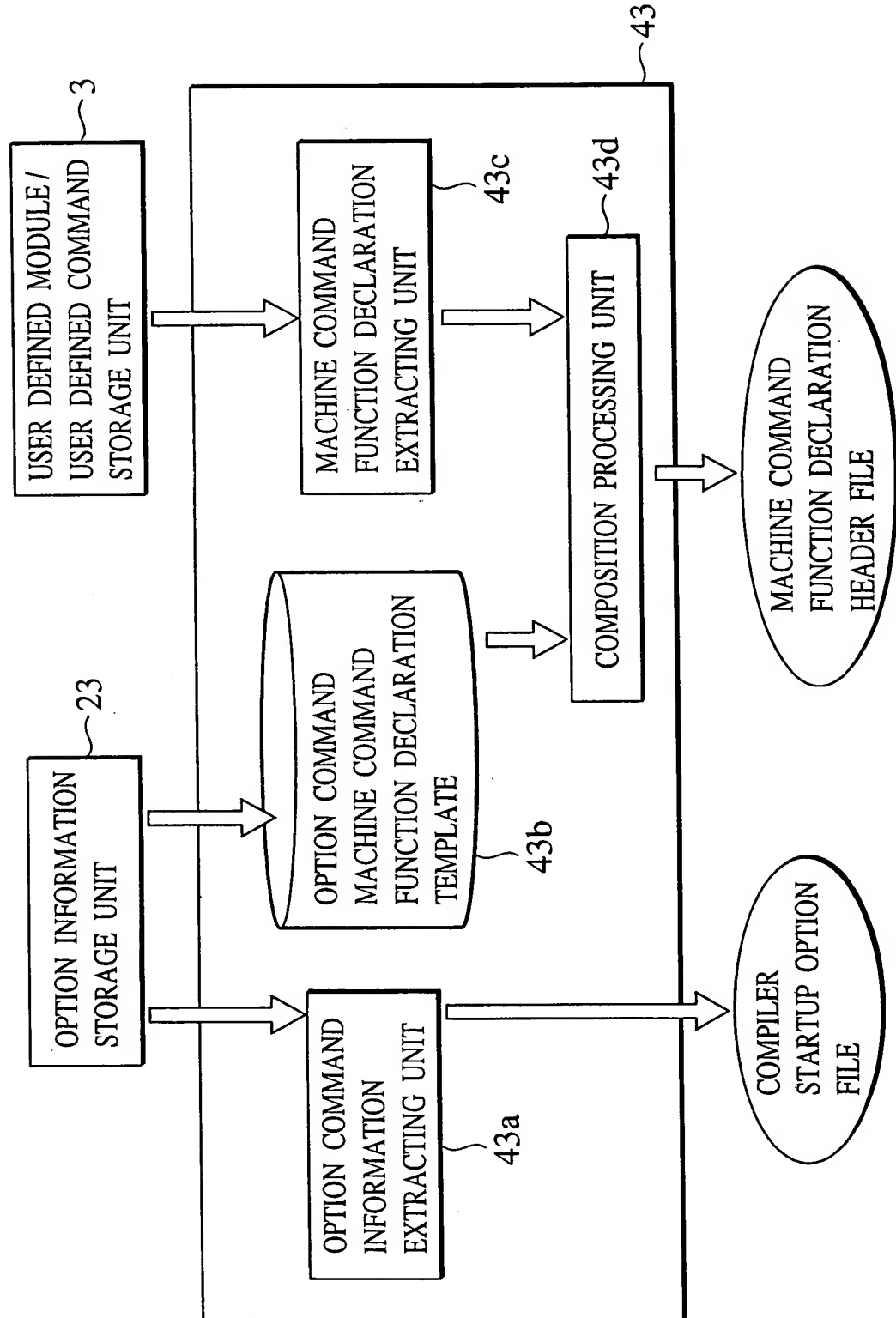


FIG.11A

-opt_minmax=ON

-opt_div=ON

-opt_bit=OFF
⋮

FIG.11B

XOR Rn, Rm {(Rn,32,U,1,1),(Rm,32,U,1,0)}
XOR1 Rn, Rm, Imm8 {(Rn,32,U,1,1),(Rm,32,U,1,0) , (Imm8,U,1,0)}

#XOR FORMAT WHEN PORTIONS OF Rn and Rm ARE ASSEMBLERS
#FOLLOWING OPERAND DEFINITION
#Rn OPERAND DISP AT(Rn,32,U,1,1) . 32 IS THE NUMBER OF BITS
#U SHOWS WHETHER A CODE IS PRESENT OR ABSENT (U IS UNSIGNED).
#THE NEXT 1 INDICATES THAT Rn IS A SOURCE OPERAND. THE LAST 1
#INDICATES THAT Rn IS A DESTINATION OPERAND

FIG.12A

```
int x=0 ;

main() {

    int i, j;
    for(i=0 ; i<10 ; i++)
        for(j=0 ; j<5 ; j++)
            x+=i*j ;
    printf("result=%d\n", x) ;
}
```

FIG.12B

```
int x=0 ;

main() {

    int i, j;
    for(i=0 ; i<10 ; i++){
        _START(1)
        for(j=0 ; j<5 ; j++)
            x+=i*j ;
        _END(1)
    }
    printf("result=%d\n", x) ;
}
```

FIG.13A

| NUMBER | START ADDRESS | END ADDRESS | NUMBER OF COMMANDS |
|--------|---------------|-------------|--------------------|
| 1 | 0x8001e | 0x80038 | 370 |

FIG.13B

| NUMBER | START ADDRESS | END ADDRESS | NUMBER OF CYCLES |
|--------|---------------|-------------|------------------|
| 1 | 0x8001e | 0x80038 | 500 |

FIG.13C

| CACHE SIZE | CACHE ERROR RATE |
|------------|------------------|
| 1Kbytes | 0.191 |
| 2Kbytes | 0.148 |
| 4Kbytes | 0.109 |
| 8Kbytes | 0.087 |
| 16Kbytes | 0.066 |

FIG.14

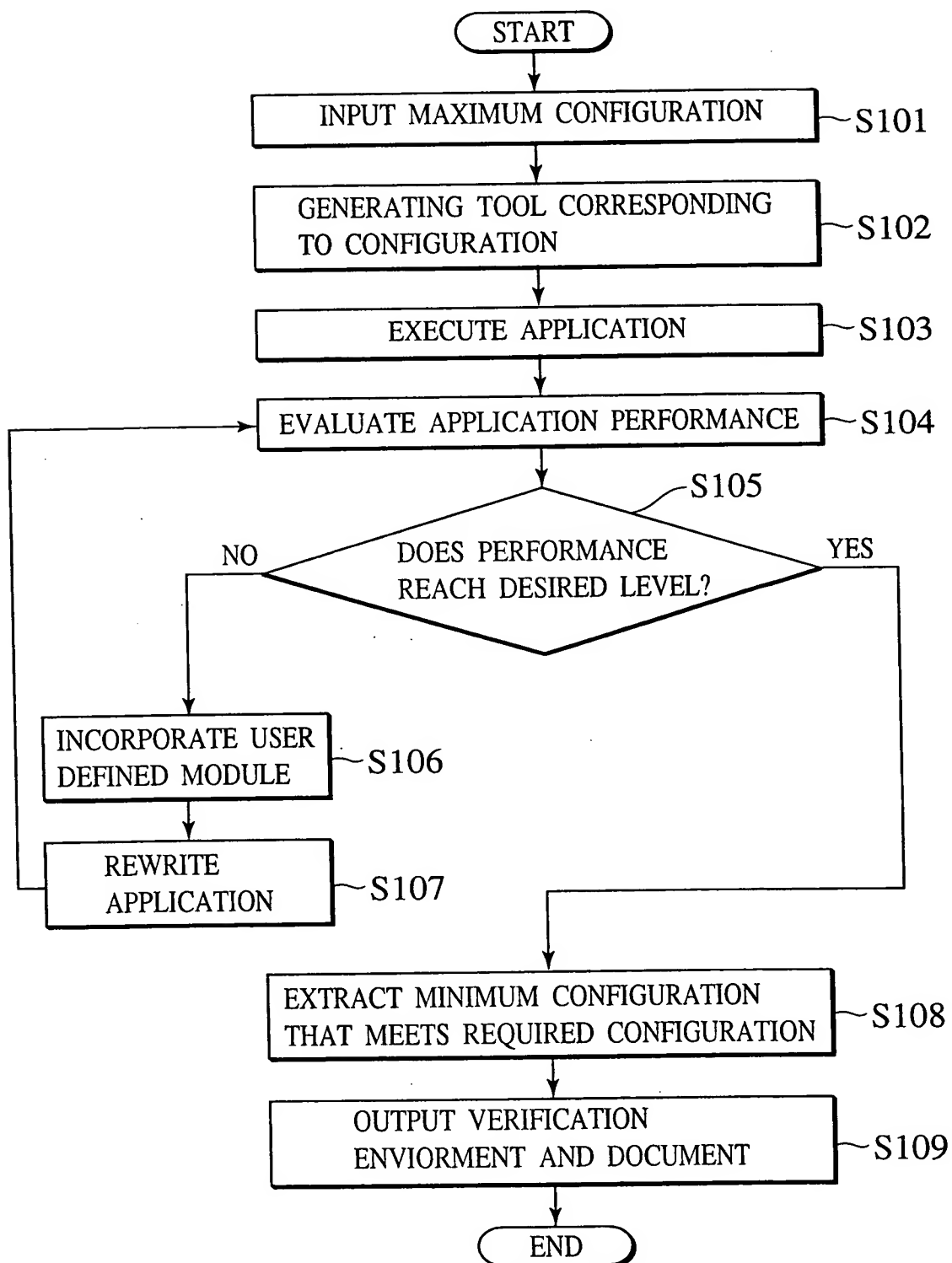


FIG.15A

```

move R5, 0x00080000
move R6, 0x00090000
move R7, 0x00080100
loop :
ld      R1, (R5)
ld      R2, 4(R5)
call    _Culc      // TO SUBROUTINE
st      R1, (R6)
add     R6, 4
add     R5, 8
not_equal R5, R7, loop //if R5 !=R7 then jump roop

```

FIG.15B

```

Move R1, 0x00080000 //SUBSTITUTE START ADDRESS INTO R1
Move R2, 0x00080100 //SUBSTITUTE START ADDRESS INTO R2
Move R3, 0x00090000 //SUBSTITUTE COMPUTATION RESULT STRAGE START LOCATION INTO R3
Move R4, 0x00000001 //SUBSTITUTE DSP START VALUE INTO R4
St_cntibus R1, (cntibus_addr1) //SPECIFY READ START LOCATION
St_cntibus R2, (cntibus_addr2) //SPECIFY READ END LOCATION
St_cntibus R3, (cntibus_addr3) //SPECIFY START LOCATION OF REWRITING COMPUTATION RESULT
St_cntibus R4, (cntibus_addr4) //START DSP PROCESSING
(COREPROCESSOR CONTINUES OTHER PROCESSING, WAIT FOR THE END OF DSP, AND CONTINUES PROCESSING)

```

FIG.15C

```

Tyedef unsigned long  address;
class DSP_HWEEngine : public =HWEEngine, public ControlBus
{
public :
virtual bool read_cntibus(address adr, unsigned long* data) ;
virtual bool write_cntibus(address adr, unsigned long data) ;
virtual bool do_command(unsigned long s1, unsigned long s2, unsigned long operand,
                        unsined long* ret) ;
}

```

FIG.16

```
Bool DSP_HWEngine ::read_cntibus(address adr, unsigned long* data)
{
    *date=cntibus[adr];
    return true ;
}
bool DSP_HWEngine ::write_cntibus(address adr, unsigned long data)
{
    if(adr==cntibus_addr4)//VALUE IS WRITTEN IN DSP CONTROL REGISTER
    {
        PROCESSING IS PERFORMED FOR WRITEN VALUE
        return true ;
    }
    cntibus[adr]=data
    return true ;
}

bool DSP_HWEngine :: do_command(unsigned long s1,unsigned long s2, unsigned long operand,
unsigned long* ret)
{
    if(operand&0x0000ff00)==0x60)
    {
        PROCESSING OF COMMAND CODE 0x60 IS WRITTEN, COMMAND SOURCE IS
        DO_COMMAND FUNCTION AUGMENT, A1 AND A2, DESTINATION IS "RET"
        THE COMPUTATION RESULT IS PLACED IN "RET", AND IS RETURNED
        *RET=RESULT
        return true ;
    }
    WHEN A USER DEFINED COMMAND IS PRESENT,"IF" ARE ARRANGED IN PLURALITY
    Else if( )
    {

    }
}
```


FIG.17A

```
Class CORE
{
public :
    void one_step_execute();
};
```

FIG.17B

```
Core1_config_set()
{
    //CORE1 OBJECT GENERATION
    //CORE1 OPTION COMMAND SETTING
    //SETTING OF HARDWARE ENGINE AND THE LIKE
    //OTHER SETTINGS(CACHE OR COPROCESSOR)
}
Core2_config_set()
{
    //CORE2 OBJECT GENERATION
    //CORE2 OPTION COMMAND SETTING
    //SETTING OF HARDWARE ENGINE AND THE LIKE
    //OTHER SETTINGS(CACHE OR COPROCESSOR)
}
Core3_config_set()
{
    //CORE3 OBJECT GENERATION
    //CORE3 OPTION COMMAND SETTING
    //SETTING OF HARDWARE ENGINE AND THE LIKE
    //OTHER SETTINGS(CACHE OR COPROCESSOR)
}
```

FIG.18A

```
Module_set()
{
    //CALL SET FUNCTION OF EACH CORE AND GENERATE AND SET OBJECT
    Core1_config_set();
    Core2_config_set();
    Core3_config_set();
}
```

FIG.18B

```
One_set()
{
    Core1>one_step_execute();
    Core2>one_step_execute();
    Core3>one_step_execute();
}
```

FIG.19

